APPLICATION FOR UNITED STATES LETTERS PATENT

FOR

METHOD FOR COMMUNICATING OCCURRENCE OF EVENTS IN A STORAGE MEDIUM

Inventor(s): Alok Sinha

Prepared by: Kenneth M. Seddon,
Patent Attorney

intel®
Intel Corporation
5000 W. Chandler Blvd., CH6-404
Chandler, AZ 85226-3699
Phone: (480) 554-9732
Facsimile: (480) 554-7738

"Express Mail" label number EL034433703US

METHOD FOR COMMUNICATING OCCURRENCE OF EVENTS IN A STORAGE MEDIUM

BACKGROUND

A redundant array of inexpensive disks (RAID) process refers to a process by which a computing system may store data throughout a storage medium, such as an array of disk drives, so that at least portions of the data may be recovered should part of the storage medium fail. A processor in such a computing system may execute algorithms that determine the appropriate parity bit and location for the data to be stored. However, in large computing systems that frequently store data to disk drives, the burden on this processor to perform the RAID process may be significant. Because of this burden, the computing system may have reduced computing resources available for other tasks.

To reduce the burden on the processor, Intelligent Input/Output (I_2O) controllers have been developed. Simply stated, I_2O processors off-load responsibility for storing data on the disk drives from the processor so that the processor is available to perform other functions. Examples of implementing the I_2O functions are described in the Intelligent Input/Output Architecture Specification, revision 1.5, published March 1997 by the I_2O Special interest Group (see http://www.i2osig.org).

Nonetheless, there is a continuing need for better ways to improve the interaction between applications running on the processor and the disk drives in a computing system that is implementing a RAID data storage process.

BRIEF DESCRIPTION OF THE DRAWINGS

The subject matter regarded as the invention is particularly pointed out and distinctly claimed in the concluding portion of the specification. The invention, however, both as to organization and method of operation, together with objects, features, and advantages thereof, may best be understood by reference to the following detailed description when read with the accompanying drawings in which:

FIG. 1 is a block diagram representation of a computing system in accordance with an embodiment of the present invention;

FIG. 2 is a block diagram representation of communicative coupling between in accordance with an embodiment of the present invention; and

FIG. 3 is a flow chart illustrating the process by which an application may register itself with an event interface in accordance with an embodiment of the present invention.

Where considered appropriate, reference numerals have been repeated among the figures to indicate corresponding or analogous elements.

DETAILED DESCRIPTION

In the following detailed description, numerous specific details are set forth in order to provide a thorough understanding of the invention. However, it will be understood by those skilled in the art that the present invention may be practiced without these specific details. In other instances, well-known methods, procedures, components and circuits have not been described in detail so as not to obscure the present invention.

Some portions of the detailed description which follow are presented in terms of algorithms and symbolic representations of operations on data bits or binary digital signals

within a computer memory. These algorithmic descriptions and representations are the techniques used by those skilled in the data processing arts to convey the substance of their work to others skilled in the art.

An algorithm is here, and generally, considered to be a self-consistent sequence of acts or operations leading to a desired result. These include physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers or the like. It should be understood, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

Unless specifically stated otherwise, as apparent from the following discussions, it is appreciated that throughout the specification discussions utilizing terms such as "processing" or "computing" or calculating" or "determining" or the like, refer to the action and processes of a computer or computing system, or similar electronic computing device, that manipulate and transform data represented as physical (electronic) quantities within the computing system's registers and/or memories into other data similarly represented as physical quantities within the computing system's memories, registers or other such information storage, transmission or display devices.

Embodiments of the present invention may include apparatuses for performing the operations herein. This apparatus may be specially constructed for the desired purposes, or it may comprise a general purpose computer selectively activated or reconfigured by a

computer program stored in the computer. Such a computer program may be stored in a computer readable storage medium, such as, but is not limited to, any type of disk including floppy disks, optical disks, CD-ROMs, magnetic-optical disks, read-only memories (ROMs), random access memories (RAMs), electrically programmable read-only memories (EPROMs), electrically erasable and programmable read only memories (EEPROMs), magnetic or optical cards, or any other type of media suitable for storing electronic instructions, and each coupled to a computer system bus.

The algorithms and displays presented herein are not inherently related to any particular computer or other apparatus. Various general purpose systems may be used with programs in accordance with the teachings herein, or it may prove convenient to construct a more specialized apparatus to perform the desired method. The desired structure for a variety of these systems will appear from the description below. In addition, embodiments of the present invention are not described with reference to any particular programming language. It will be appreciated that a variety of programming languages may be used to implement the teachings of the invention as described herein.

Referring to FIG. 1, a processor-based, computing system 10 may include a processor 12 coupled to an interface 14. Although the scope of the present invention is not limited so as to require interface 14, interface 14 may be a bridge or a part of a chipset, for example. Interface 14 may be coupled to a graphics controller 18, for example, by an accelerated graphics port (AGP) bus 16 (see Accelerated Graphics Port Interface Specification, Rev. 1.0, published on July 31, 1996 by Intel Corporation, Santa Clara, California). Controller 18 may control a display 20 to display information generated and processed by processor 12, for example. Interface 14 may also be coupled to a host

system memory 21. Host system memory 21, for example a disk drive, cache memory, or static random access memory (SRAM), may be a machine-readable storage medium that may be used to provide the instructions executed by processor 12.

Interface 14 may also be coupled to a bus 22. Although the scope of the present invention is not limited in this respect, in one embodiment, bus 22 may comprise a Peripheral Component Interconnect (PCI) bus which is compliant with the PCI Local Bus Specification, Rev. 2.1, dated June 1, 1995 and available from the PCI Special Interest Group, Portland, OR 97214.

Bus 22 may also optionally be coupled to an interface 24, which may be part of a chipset or which may be implemented by a bridge, for example, although the scope of the present invention is not limited in this respect. Also coupled to bus 22, may be an input/output (I/O) device 26, which may be a host adapter in one embodiment of the invention. I/O device 26 may be, for example, a SCSI Fibre Channel (ANSI Standard X.3230-1994-Fibre Channel Physical and Signaling Standard (FC-PH) available at http://www.fibrecannel.com/technology/tech-frame.htm), an Ethernet device, or a Next Generation I/O (NGIO) device (available at http://www.clients.activate.net.telspan/ngioforum/072199.htm).

Computing system 10 may also include an Intelligent Input/Output (I_2O) device 30, which in turn, may be coupled to one or more disk drives 35. Although not limited in scope in this respect, I_2O device 30 may optionally include components such as non-volatile memory, an SCSI controller, and a processor such as the i960®Rx I/O processor available from Intel Corporation, Santa Clara, CA. In the embodiment illustrated in FIG. 1, I_2O device 30 is coupled to a storage medium, such as disk drive(s) 35, which represent

any storage medium that may be used to implement a redundant array of inexpensive disks (RAID) data storage process. Although the scope of the present invention is not limited in this respect, in this embodiment I_2O device 30 is intended to perform at least some of the instructions used to store and retrieve data in a RAID system so as to reduce the burden on processor 12 of system 10. Computing system 10 may also include other modules or components 32 as desired.

During the operation of system 10, situations may occur in either hardware or software that prompt system 10 to respond accordingly. These situations may occur either externally to computing system 10, such as in peripherals like storage mediums, or the situations may occur internally in the software operating within computing system 10. These situations are referred to as events. Although the scope of the present invention is not limited in this respect, examples of events that may occur in a computing system using a RAID process to store data include: notification that a disk drive has failed or is about to fail, a change in the spin rate or temperature of a disk drive, expiration of the predicted lifetime of a disk drive, array migrations, initialization of disk drives, redistribution of data after a drive failure, recovery of data from remaining disks upon failure of one or more disks, start or end of a disk format, an array expansion by adding more disks, or deletion or creation of a raid volume.

Again, the term "event" need not be limited to situations associated with a disk drive in a RAID system and the type of events to be reported are not meant to be limited to those described above. In alternative embodiments of the present invention, the occurrence of other types of "events" may be communicated such as ones associated with, for illustration only, peripheral communications, security, data processing, power

management, etc.

Referring now to FIG. 2, an embodiment that provides for communicating the occurrence of an event to other applications is provided. In the embodiment illustrated in FIG. 2, each of the various components is implemented as a software routine, program, or a dynamic linked library (DLL). However, it should be understood that in other embodiments of the present invention, various modules shown in FIG. 2 may be implemented as hardware, for example logic circuits, programmable logic arrays (PLA's), or the like. Furthermore, it should be understood that the scope of the present invention is not limited to computing systems that operate each and every module shown in FIG. 2. For example, alternative embodiments may not perform all of the modules shown, or some of the modules may be performed by other computing systems (not shown). Moreover, some of the functions shown in FIG. 2 may be combined into one computer program or separated into several independently running programs.

FIG. 2 is a block diagram representation of various modules that may be operating on processors within or external to computing system 10. The illustration is intended to demonstration how the various modules are communicatively coupled to each other in this particular embodiment while computing system 10 is in operation. The following embodiment is described as it may be implemented in a Windows NT ® platform. Windows NT ® is a registered trademark of Microsoft Corporation, Redmond, WA. However, the present invention may also be implemented on other software platforms, such as Linux, UnixWare, or the like.

FIG. 2 includes a Windows NT® Operating System Services Module (OSM) 50, which may be used to manage the interaction between the operating system running on

computing system 10 and external peripherals via bus 22. Operating above Windows NT® OSM 50 may be a RAID monitor service program 51 that coordinates the RAID process between a host processor (e.g., processor 12) and an I_2O controller (e.g., I_2O device 30). Among other things, RAID monitor service 51 may record, track, and report events that are detected by I_2O controllers or other peripheral controllers.

Abstracted from, and residing above, RAID monitor service 51, is an event application programming interface (API) 55 which may be used to notify one or more desktop management applications 60 (hereinafter, management applications or applications) of the occurrence of an event(s). However, the scope of the present invention is not limited to this particular event API as alternative programming interfaces may also be used. Management application(s) 60 may be programs, routines, algorithms, DLL's, or the like that are being executed by the host processor. Examples of management applications include: Simple Network and Management Protocol Extension Agents (SNMP's), Desktop Management Interface Component Instrumentation (DMI's), RAID monitor applications, or like. However, it should be understood that the scope of the present invention is not restricted in this respect as the embodiments of the present invention may be used to communicate the occurrence of virtually any event to any application that requests such information.

Referring to FIG. 3, an embodiment of a process for registering a management application 60 with an event API 55 is now provided. At box 100, management application 60 provides event API 55 with registration parameters associated with an event. For example, management application 60 may provide information that identifies by name or numeric value the I/O processor, the storage medium, or the type of event

that event API 55 is to notify management application 60 of. It should also be understood that the values or identification information provided may be optionally coded, if desired.

Management application 60 may also provide event API 55 with a callback function. The callback function may identify the response to be taken by either management application 60 or another program running on computer system 10. For example, the callback function may identify the routine or DLL to be executed once event API 55 notifies management application 60 of the occurrence of the event. Although the scope of the present invention is not limited in this respect, management application 60 may provide this information to event API 55 in the form of a data structure that includes fields associated with each piece of event information.

Upon receipt, event API 55 determines if the parameters are valid, box 101. If the registration request does not contain valid information, then event API 55 may return a NULL character, box 102, to indicate that registration has not occurred. If the request is valid, then event API 55 may allocate memory and store the data or the data structure associated with the event that event API 55 is to notify management application 60 of, box 103.

Once the data has been stored, event API 55 establishes a connection (e.g., an interprocess communication) with RAID monitor service 51 through a named pipe. Event API 55 may store the handle as part of the data associated with the event to be reported, box 104. As explained in more detail below, the handle data may later be used by management application 60 to unregister itself with event API 55.

Once registered, event API 55 monitors the events, including hardware events, that

occur within computing system 10. This includes the events that are reported through Window NT® OSM 50 and RAID monitor service 51. During the operation of computing system 10, RAID monitor service 51 may constantly or periodically update event API 55 with events that have recently occurred. If an event occurs that is of the type previously registered, event API 55 may use the callback function information to notify management application 60 of the occurrence of the event. Event API 55 may also provide management application 60 with any data associated with the event or the context that was originally specified by management application 60 when it registered with event API 55.

If event API 55 has notified management application 60 of the occurrence of a particular event, or if management application 60 no longer desires to be notified of a particular event, management application 60 may unregister itself with event API 55. To do so, management application 60 may simply pass the handle associated with the registration of this particular event, see box 104 of FIG. 2, so that event API 55 will proceed as if that handle was invalid.

In the embodiments described above, management application 60 registered a single event with event API 55. However, it should be understood that the scope of the present invention is not limited in this respect as different management applications may register with the event API, each to be notified of the same event, or one management application may register several events that the event API is to notify that management application once those events occur.

One advantage of the embodiments of the present invention is that desktop management applications may now be notified of the occurrence of events without having

to actively search for their occurrence. This improves the efficiency of the management applications as they no longer have to randomly or periodically poll the RAID monitor service to check if an event has occurred. This also improves the response time of the management application as the management application may be notified of the occurrence of an event as soon as the event API becomes of aware of it.

The present invention also provides the management application with information that was previously unavailable and not supported on Windows NT® platforms. Thus, the functionality and flexibility of the management applications may be improved. Furthermore, embodiments of the present invention also obviate the need to modify each management application should there be a change in the event information provided by a RAID system. For example, if the format for reporting events by the RAID process should change, only the event API may need be updated to reflect any changes. It is not necessary that each management application be modified since the event API abstracts the event reporting function. This improves the compatibility of the applications that may run within a computing system and provides for seamless transitions if the RAID process is modified.

While certain features of the invention have been illustrated and described herein, many modifications, substitutions, changes, and equivalents will now occur to those skilled in the art. It is, therefore, to be understood that the appended claims are intended to cover all such modifications and changes as fall within the true spirit of the invention.